Apple Assembly Line

$1.80

Volume 4 -- Issue 4 January, 1984

In This Issue...

News from Apple

Apple sent us a mouse the other day, and we hope to build some software around it. The mouse came with a disk of graphic software (done by Bill Budge) which makes the plain old Apple II look almost as good as Lisa. I didn't know your could do all that on a 280x192 screen, and as fast as he does it. The mouse itself appears identical to the Lisa mouse. It attaches to a cute red interface card you plug into any slot. The card has a version of the 6805 microprocessor on it...the kind with internal ROM and RAM which is not visible from the outside world.

Apple also is spreading the word that future Apple //e's are going to have 32 icon characters in the alternate character set. This probably means some changes to the Cx ROM... (Erv Edge says he hopes that means they are going to fix some bugs, too!)

Another tidbit from Apple is that future //e's will have most of the chips soldered to the motherboard, rather than riding in sockets. They say that should solve most of the remaining reliability problems. In my experience, Apple doesn't have any reliability problems. And I like sockets, because I like to tinker. And if something eventually does go bad, it is certainly easier to trade chips than motherboards. Nevertheless, they have made up their minds.

Profiler.....Bill Morgan

For the last several months, I've been intrigued by an article in the August '83 issue of Byte Magazine, "Chisel Your Code with a Profiler", by Dennis Leas and Paul Wintz. They describe a utility program, called a profiler, which measures where an executing program is spending most of its time. The largest application for such a tool is testing programs compiled from a high-level language. Typically such a program will spend nearly all of its time executing only a small section of the code. Leas and Wintz claim that the proportion is about 90% of the time in about 10% of the code. With a profiler you can identify the bottleneck and speed up the whole program by recoding one small piece.

The profiler first divides your program into sixteen "bins". It then interrupts your program periodically and reads the stored Program Counter from the stack. If the program is in the area you want to measure, it increments one of an array of counters. The profiler then returns control to your program until the next interrupt occurs. When the testing period is finished you can display the counters and spot your problem areas.

An essential part of this tool is a source of regularly timed interrupts. The best place to get a timed interrupt signal is from a suitable clock card. All of the clock cards have some provision for generating interrupts, usually at intervals of about 1 millisecond or 1 second. Some also have available 64 Hz or 256 Hz frequencies, or other values. Check the documentation with your clock to see exactly how to use its interrupt features.

The interrupt timing you want to use will in part be a function of how long your program, or subroutine, will run. If you're profiling a sort that takes several minutes to complete, a 1000 Hz interrupt will overflow the counters long before a significant amount has been done. If the routine takes a short time, a 60 Hz clock won't catch enough hits to be meaningful. Leas and Wintz use a 6 Hz signal picked up from their disk drives to profile a compiler that runs for about 10 minutes.

If all you have available is a high-frequency signal, it's easy enough to divide it down to something usable. Just initialize a counter in the setup portion of the program to the necessary value. Then whenever an interrupt occurs, decrement the counter. Most of the time the counter will be non-zero, so then branch directly to the exit portion of the handler. When the counter reaches zero, go ahead and do the full interrupt processing and then reset the counter.

What if I don't have a clock?, you ask. That is exactly the problem I had when I started thinking about this project. Then I ran across an article in the July 83 issue of Micro in which Charles Putney (a subscriber and sometimes contributor to these pages) told how to get a 60 Hz signal to the interrupt line. Charles' article tells how to use that signal to implement a real-time clock, but it seemed to me that here was exactly the

S-C Macro Assembler Version 1.0.....$80.00
S-C Macro Assembler Version 1.1 Update.....$12.50
Full Screen Editor for S-C Macro Assembler..... $49.00
    Includes complete source code.
S-C Cross Reference Utility.....$20.00
S-C Cross Reference Utility with Complete Source Code.....$50.00
DISASM Dis-Assembler (RAK-Ware).....$30.00
Quick-Trace (Anthro-Digital).....(reg. $50.00) $45.00
The Visible Computer: 6502 (Software Masters).....(reg. $50.00) $45.00

S-C Word Processor (the one we use!).....$50.00
    With fully commented source code.
Applesoft Source Code on Disk.....$50.00
    Very heavily commented. Requires Applesoft and S-C Assembler.
ES-CAPE: Extended S-C Applesoft Program Editor.....$60.00

AAL Quarterly Disks.....each $15.00
    Each disk contains all the source code from three issues of "Apple Assembly Line", to save you lots of typing and testing time.

QD#1: Oct-Dec 1980	QD#2: Jan-Mar 1981	QD#3: Apr-Jun 1981
QD#4: Jul-Sep 1981	QD#5: Oct-Dec 1981	QD#6: Jan-Mar 1982
QD#7: Apr-Jun 1982	QD#8: Jul-Sep 1982	QD#9: Oct-Dec 1982
QD#10: Jan-Mar 1983	QD#11: Apr-Jun 1983	QD#12: Jul-Sep 1983
QD#13: Oct-Dec 1983		

Double Precision Floating Point for Applesoft.....$50.00
    Provides 21-digit precision for Applesoft programs.
    Includes sample Applesoft subroutines for standard math functions.
Amper-Magic (Anthro-Digital).....(reg. $75.00) $67.50
Amper-Magic Volume 2 (Anthro-Digital).....(reg. $35.00) $30.00
Routine Machine (Southwestern Data Systems).....(reg. $64.95) $60.00
FLASH! Integer BASIC Compiler (Laumer Research).....$79.00
Fontrix (Data Transforms)..... $75.00
Aztec C Compiler System (Manx Software).....(reg. $199.00) $180.00

Blank Diskettes.....package of 20 for $45.00
    (Premium quality, single-sided, double density, with hub rings)
Vinyl disk pages, 6"x8.5", hold one disk each.....10 for $6.00
Diskette Mailing Protectors.....10-99: 40 cents each
                100 or more: 25 cents each
   Cardboard folders designed to fit into 6"X9" Envelopes.
Envelopes for Diskette Mailers..... 5 cents each
ZIF Game Socket Extender.....$20.00

Grappler+ Printer Interface (Orange Micro).....($175.00) $150.00
Bufferboard 16K Buffer for Grappler (Orange Micro).....($175.00) $150.00
Buffered Grappler+ NEW!! Interface and 16K Buffer.....($239.00) $200.00

Books, Books, Books.....compare our discount prices!

"The Apple][Circuit Description", Gayler	($22.95)	$21.00
"Understanding the Apple II", Sather	($22.95)	$21.00
"Enhancing Your Apple II, vol. 1", Lancaster	($17.95)	$17.00
"Incredible Secret Money Machine", Lancaster	($7.95)	$7.50
"Beneath Apple DOS", Worth & Lechner	($19.95)	$18.00
"Bag of Tricks", Worth & Lechner, with diskette	($39.95)	$36.00
"Assembly Lines: The Book", Roger Wagner	($19.95)	$18.00
"What's Where in the Apple", Second Edition	($24.95)	$23.00
"What's Where Guide" (updates first edition)	($9.95)	$9.00
"6502 Assembly Language Programming", Leventhal	($18.95)	$18.00
"6502 Subroutines", Leventhal	($17.95)	$17.00

  Add $1.50 per book for US shipping. Foreign orders add postage needed.

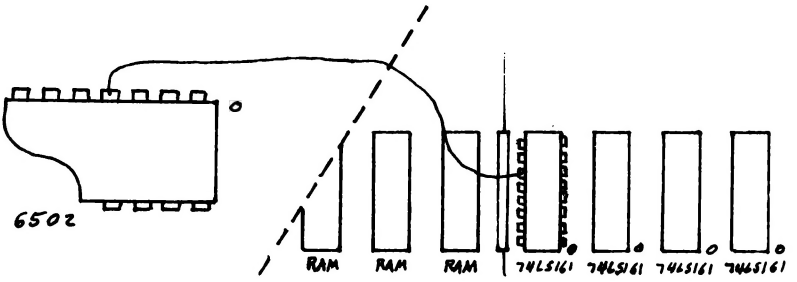
Whatever Else You Need.....Call for Our Low Prices

        *** S-C SOFTWARE, P. O. BOX 280300, Dallas, TX 75228 ***
        ***   (214) 324-2050   ***
        *** We accept Master Card, VISA and American Express ***

interrupt signal I had been seeking for my profiler.

All you need to do is add a wire inside your Apple, from pin 11 of the 74LS161 at coordinate D11 to pin 4 of the 6502. I used a pair of plunger clips (Radio Shack #270-370, the smallest ones) to attach the wires, and also put a pushbutton in the circuit. When attaching the clips to the IC pins, be EXTREMELY careful not to short any adjacent pins, and try to arrange things so that the wire doesn't wobble around. TURN THE POWER OFF BEFORE MESSING WITH WIRING INSIDE YOUR COMPUTER.

Here's a drawing that shows where to connect the wires:



Note that the photograph in the Micro article does NOT show the correct pins. The description in Putney's text is correct, but whoever did the photo artwork garbled it.

The signal we are borrowing is one of the video timing signals, called V5. V5 is normally high (~5 volts). It goes low every 1/60th of a second, and stays low for about 380 microseconds. That's a pretty good interrupt signal, but we're going to have to allow time for V5 to get back to its high state before we return to the main program, or we'll get more than one interrupt per cycle.

The program

When you BRUN or CALL Profiler, lines 1120-1130 hook the Initialize portion of the program into the monitor's CTRL-Y vector.

Initialize first connects the Handler routine to the IRQ vector (1190-1220). It then gets the starting and ending addresses from where the Monitor left them, takes the difference and divides by 16 to get the size of each bin (1270-1390).

Build Table then starts the table with the Start address and loops to set each table entry Step bytes larger than the previous one (1420-1650). At the same time the routine sets the count for each bin to zero and adds an extra zero byte after the count (1610-1630). This extra byte makes the table easier to read with a Monitor memory dump.

Note that the last entry is set to the End value, rather than a

calculated step (1670-1700). This makes the last bin larger than the others, by somewhere between 0 and 15 bytes, to compensate for the remainder left behind when we divided to get Step.

Now we come to the Handler itself. When an IRQ interrupt occurs we first save all the registers on the stack (1740-1790). The next step is to extract the Program Counter value from inside the stack and save it (1800-1840).

The next step is checking that PC value to see if it is inside the range we want (1860-1920). If not, go on to Exit.

If we are in range, search down the table to find the right bin (1930-1980) and register the count (2000-2040). Since the counters only go up to 255, I have the profiler stop when one of them wraps around (2070-2080).

The Exit routine includes a delay loop (2120-2140) to make sure that the Handler takes at least 380 microseconds. This insures that the V5 line we're using for an interrupt source has gone back high, and won't interrupt again as soon as the RTI is done. If you're lucky enough to be getting your interrupts from a clock card you won't need this loop, but you will need to do something to tell the card that you're done with this interrupt. Check your clock manual. Profiler then ends by restoring the registers and doing an RTI.

The Compare Entry routine (2220-2270) just compares the PC value to the current table value.

The funny-looking FILLER space (line 2340) makes sure that the Table begins on a new line in the Monitor memory dump, keeping things easy to read.

Using Profiler

When I want to profile a program, I first assemble Profiler to run somewhere out of the way above or below the program I want to test. Then I enter the Monitor and type addrG to connect Profiler to CTRL-Y. Next I enter addr1.addr2^Y (that's <Start-address>.<End-address><CTRL-Y>) to initialize things.

The next step is to start the program I want to measure, and then start the interrupts coming. My system has a pushbutton between the 60 Hz source and the IRQ line, so I just hold the button down for the period I want to check on. If you're using a clock card you can probably insert instructions into your program to start and stop the interrupts at the points you want.

If one of the counters passes 255 Profiler will Break into the Monitor. Otherwise get into the monitor after your program has finished and examine the table. There's a record of exactly where your program has been.

In a large program, the bin with the highest count may be too

wide to really tell where the bottleneck is. If so, just use the control Y command to profile only the bin that had the largest count. This will divide that section into 16 segments so you can see more detail.

Limitations and possible improvements

The profiler described by Leas and Wintz displays the counts as a bar graph, so the largest count really stands out. My version just leaves the addresses and counts where you can read them with the Monitor, so I'm sure you can come up with ways to improve that.

Sometimes it would be nice to be able to build the address table interactively, rather than having it forced to sixteen equal-sized sections. Maybe something like entering the starting address you want for each bin, and a zero at the end.

The DOS problem

There has always been a problem with using interrupts in the Apple II under DOS 3.3, but the solutions are now pretty well-known. Elsewhere in this issue we cover the DOS or Monitor patches necessary to use the 6502's IRQ interrupt without trouble. This program assumes that all that has been taken care of, or that you don't care.

References

1. "Chisel Your Code with a Profiler" Dennis Leas & Paul Wintz. Byte Magazine. August, 1983, pp 286-290.
2. "A Clock Interrupt for Your Apple" Charles Putney. Micro Magazine. July, 1983, pp 36-41.

```
                    1000 *SAVE S.PROFILER
                    1010 *-------------------------------
003E-               1020 A2L     .EQ $3E
003F-               1030 A2H     .EQ $3F
0040-               1040 A3L     .EQ $40
0041-               1050 A3H     .EQ $41
                    1060
0100-               1070 STACK       .EQ $100
03FE-               1080 IRQ.VECTOR  .EQ $3FE
03F9-               1090 CONTROL.Y.VECTOR .EQ $3F9
                    1100 *-------------------------------
                    1110         .TF PROFILER
0800- A9 08         1120         LDA /INITIALIZE
0802- 8D FA 03      1130         STA CONTROL.Y.VECTOR+1
0805- A9 0B         1140         LDA #INITIALIZE
0807- 8D F9 03      1150         STA CONTROL.Y.VECTOR
080A- 60            1160         RTS
                    1170 *-------------------------------
                    1180 INITIALIZE
080B- A9 79         1190         LDA #HANDLER       install vector
080D- 8D FE 03      1200         STA IRQ.VECTOR
0810- A9 08         1210         LDA /HANDLER
0812- 8D FF 03      1220         STA IRQ.VECTOR+1
0815- A9 00         1230         LDA #0         initialize variables
0817- 8D CF 08      1240         STA HITS
081A- 8D D0 08      1250         STA HITS+1
```

QUICKTRACE

relocatable program traces and displays the actual machine operations, *while* it is running without interfering with those operations. Look at these *FEATURES:*

***Single-Step** mode displays the last instruction, next instruction, registers, flags, stack contents, and six user-definable memory locations.*

***Trace** mode gives a running display of the Single-Step information and can be made to stop upon encountering any of nine user-definable conditions.*

***Background** mode permits tracing with no display until it is desired. Debugged routines run at near normal speed until one of the stopping conditions is met, which causes the program to return to Single-Step.*

***QUICKTRACE** allows changes to the stack, registers, stopping conditions, addresses to be displayed, and output destinations for all this information. All this can be done in Single-Step mode while running.*

***Two optional display formats** can show a sequence of operations at once. Usually, the information is given in four lines at the bottom of the screen.*

***QUICKTRACE** is completely transparent to the program being traced. It will not interfere with the stack, program, or I/O.*

***QUICKTRACE** is relocatable to any free part of memory. Its output can be sent to any slot or to the screen.*

***QUICKTRACE** is completely compatible with programs using Applesoft and Integer BASICs, graphics, and DOS. (Time dependent DOS operations can be bypassed.) It will display the graphics on the screen while **QUICKTRACE** is alive.*

***QUICKTRACE** is a beautiful way to show the incredibly complex sequence of operations that a computer goes through in executing a program*

QUICKTRACE *$50*

Is a trademark of Anthro-Digital, Inc.

Written by John Rogers

See these programs at participating Computerland and other fine computer stores.

Anthro - Digital Software, Inc.
P.O. Box 1385 Pittsfield, MA 01202

```
081D- 38       1260         SEC
081E- A5 3E    1270         LDA A2L
0820- E5 40    1280         SBC A3L       calculate step size
0822- 8D D1 08 1290         STA STEP
0825- A5 3F    1300         LDA A2H
0827- E5 41    1310         SBC A3H
0829- 90 4D    1320         BCC ERROR     end<start
082B- 8D D2 08 1330         STA STEP+1
               1340
082E- A2 03    1350         LDX #3        divide STEP by 16
0830- 4E D2 08 1360 .1      LSR STEP+1    (shift it right 4)
0833- 6E D1 08 1370         ROR STEP
0836- CA       1380         DEX
0837- 10 F7    1390         BPL .1
               1400
               1410 BUILD.TABLE
0839- A5 40    1420         LDA A3L        first table entry
083B- 8D D8 08 1430         STA TABLE      is start address
083E- A5 41    1440         LDA A3H
0840- 8D D9 08 1450         STA TABLE+1
0843- A2 00    1460         LDX #0
0845- 8E DA 08 1470         STX TABLE+2    zero count
0848- 8E DB 08 1480         STX TABLE+3    and fill byte
               1490
084B- E8       1500 .1      INX            next entry
084C- E8       1510         INX
084D- E8       1520         INX
084E- E8       1530         INX
084F- 18       1540         CLC
0850- BD D4 08 1550         LDA TABLE-4,X
0853- 6D D1 08 1560         ADC STEP       add step size to
0856- 9D D8 08 1570         STA TABLE,X    last entry
0859- BD D5 08 1580         LDA TABLE-3,X
085C- 6D D2 08 1590         ADC STEP+1
085F- 9D D9 08 1600         STA TABLE+1,X
0862- A9 00    1610         LDA #0
0864- 9D DA 08 1620         STA TABLE+2,X zero count
0867- 9D DB 08 1630         STA TABLE+3,X and fill byte
086A- E0 3C    1640         CPX #$3C      done?
086C- 90 DD    1650         BCC .1        no
               1660
086E- A5 3E    1670         LDA A2L
0870- 9D DC 08 1680         STA TABLE+4,X and make last entry
0873- A5 3F    1690         LDA A2H       equal end
0875- 9D DD 08 1700         STA TABLE+5,X
0878- 60       1710 ERROR   RTS
               1720 *--------------------------------
               1730 HANDLER
0879- A5 45    1740         LDA $45       get A back from where
087B- 48       1750         PHA           Monitor stashed it
087C- 8A       1760         TXA
087D- 48       1770         PHA           save registers
087E- 98       1780         TYA
087F- 48       1790         PHA
0880- BA       1800         TSX
0881- BD 06 01 1810         LDA STACK+6,X     get PC from stack
0884- 8D CE 08 1820         STA PCH
0887- BD 05 01 1830         LDA STACK+5,X
088A- 8D CD 08 1840         STA PCL
               1850 *---SEARCH TABLE----------------
088D- A2 00    1860         LDX #0            compare PC to start
088F- 20 C0 08 1870         JSR COMPARE.ENTRY   of table
0892- 90 21    1880         BCC EXIT          below table
0894- A2 40    1890         LDX #$40          and compare to
0896- 20 C0 08 1900         JSR COMPARE.ENTRY   end of table
0899- B0 1A    1910         BCS EXIT          above table
               1920
089B- CA       1930 .1      DEX               next entry
089C- CA       1940         DEX
089D- CA       1950         DEX
089E- CA       1960         DEX
089F- 20 C0 08 1970         JSR COMPARE.ENTRY
08A2- 90 F7    1980         BCC .1            not there yet
               1990
08A4- EE CF 08 2000         INC HITS      count hit in total
08A7- D0 03    2010         BNE .2
08A9- EE D0 08 2020         INC HITS+1
08AC- FE DA 08 2030 .2      INC TABLE+2,X count hit in bracket
08AF- D0 04    2040         BNE EXIT
               2050
```

```
                    2060 * counter overflowed, so put it back to $FF and end
08B1- DE DA 08       2070         DEC TABLE+2,X
08B4- 00             2080         BRK
                     2090 * ... do whatever it takes to clean up the stack
                     2100 *     and display the results
                     2110
08B5- A0 37          2120 EXIT    LDY #55       delay about 275 usec so
08B7- 88             2130 .1      DEY             V5 will be high on exit
08B8- D0 FD          2140         BNE .1
08BA- 68             2150         PLA           restore registers
08BB- A8             2160         TAY
08BC- 68             2170         PLA
08BD- AA             2180         TAX
08BE- 68             2190         PLA
08BF- 40             2200         RTI           and exit
                     2210 *--------------------------------
                     2220 COMPARE.ENTRY
08C0- AD CD 08       2230         LDA PCL
08C3- DD D8 08       2240         CMP TABLE,X
08C6- AD CE 08       2250         LDA PCH
08C9- FD D9 08       2260         SBC TABLE+1,X
08CC- 60             2270         RTS
                     2280 *--------------------------------
                     2290 VARIABLES
08CD-                2300 PCL     .BS 1         program counter
08CE-                2310 PCH     .BS 1
08CF-                2320 HITS    .BS 2         total count
08D1-                2330 STEP    .BS 2         bracket size
08D3-                2340 FILLER .BS *|8*8+8-* align table
                     2350 *--------------------------------
08D8-                2360 TABLE   .EQ *
```

More from Don Lancaster.....Bill Morgan

A couple of people have pointed out to me that we have advertised Don Lancaster's "Micro Cookbook, Volume II", but have never described it. This one is subtitled Machine Language Programming, and picks up where Volume I left off. He devotes about 450 pages to machine language programming, simple I/O ports, and his Micro Applications Attack method of problem-solving.

Lancaster's method of teaching machine language looks a little strange from my perspective: he says don't even think about an assembler until you have thoroughly learned the instructions from hand assembly. He lays out a system of learning all the instructions and addressing modes by documenting them on 3X5 cards. All his examples refer to the 6502, but the system can be applied to any processor. I suppose this IS a great way to engrave into your memory exactly how a processor works. All this is handled in Don's usual entertaining and enlightening fashion.

This is a good place to mention another book of Lancaster's that has been around for a while: The Hexadecimal Chronicles. This is a huge collection of conversion tables for moving around between ASCII, decimal, hexadecimal and octal (including Apple's negative decimal way of handling addresses.) My TI Programmer calculator is a lot smaller and easier to use, but much more expensive too. If you can find a copy of this book, look it over carefully. It may be exactly what you need.

DOS Patches to Avoid Interrupt Trouble.....Bill Morgan

As we reported a couple of years ago (V2N4, Jan 82), there is a serious problem in using interrupts in the Apple. The Monitor's IRQ interrupt handler uses location $45 to store the contents of the A-register while it is checking to see if the interrupt was from IRQ, or from a BRK instruction. Unfortunately, DOS 3.3 uses $45 for temporary storage in several different routines. If an IRQ interrupt occurs while DOS is active, the Monitor clobbers $45 and DOS can lose a variable.

The usual solution has been to change the Monitor to use some other address to stash the Accumulator. This can be done by copying the Monitor into the RAM card and patching in the new address, or by burning a new Monitor in EPROM and modifying the Apple to accept the chip. The byte that needs changing is at $FA41 in the Autostart ROM, or $FA87 in the Old Monitor ROM.

In the January 84 issue of Washington Apple Pi, Bruce Field reports about the other approach to resolving the conflict. He passes along Wilton Helm's details of the locations in DOS that refer to $45, and how to change things around to safely use interrupts without affecting anything else. Here's Helm's report:

> "Location $45 is used at the following places in DOS 3.3:
>
> $A133 $A13E $A158 $A1BE $A1D3 $A1E8 $A1F7 $A1F9 $A201
> $A2CC $A767 $A77F $ADBA $AE0A $AE54 $AE58 $BED3 $BF16
> $BF39 $BF55 $BF57 $BF5B $BF9D $BFA3 $BFA5.
>
> These locations should be changed to $46. Location $46 is used for only one purpose, at $BA06 and at $BDA4. These two locations should be changed to $2C. Location $2C is used only by RWTS subroutines and does not conflict with this additional use. The end result is that DOS no longer uses $45 and does not use any new locations."

Field also reports that "these modifications have been made in Universal DOS ... and similar patches have been made in Diversi-DOS."

Bob S-C put together the following Applesoft program to install the patches. The program first checks to make sure that the DOS in memory has not had the patches applied already, then puts them in place. The check beforehand will also avoid clobbering a non-standard DOS.

```
100  REM PREPARE DOS 3.3 FOR INTERRUPTS
110  READ A: IF A = 0 THEN 200
120  IF  PEEK (A) = 69 THEN 110
130  PRINT "THIS DOS IS ALREADY PATCHED": END
140 I = I + 2: GOTO 120
200  READ A: IF A = 0 THEN 300
210  IF  PEEK (A) = 70 THEN 200
```

```
220  PRINT "THIS DOS IS ALREADY PATCHED": END
300  RESTORE
310  READ A: IF A <  > 0 THEN  POKE A,70: GOTO 310
320  READ A: IF A <  > 0 THEN  POKE A,44: GOTO 320
330  END
1000  DATA  41267,41278,41304,41406,41427,41448,41463,41465,
     41473,41676,42855,42879,44474,44554,44628,44632,48851,
     48918,48953,48981,48983,48987,49053,49059,49061,0
1010  DATA  47622,48548,0
```

While we're on the subject of interrupts, I'd like to recommend a book to you: "Real Time Programming - Neglected Topics", by Caxton C. Foster. (Addison-Wesley, 1981. Paperback, $8.95 a couple of years ago.) Foster covers interrupts, ports, timing considerations, A/D conversion, filters, control loops, and communication issues. He points out that these are "enough topics to make up the better part of a full-fledged masters program in electrical engineering or computer science" and that "no book less than 10 inches thick could cover all these topics in detail." Nevertheless, in about 180 pages he does an excellent job of introducing the reader to the material, covering both hardware and software.

It Was a Bad Dream, I Think.....Bob Sander-Cederlof

For two hours two nights ago I tossed, turned, wrestled, and wrote a speech on trends in our favorite industry. I think it went like this....

3-piece Suits

Woz likes blue jeans and jogging shoes. Engineers and programmers tend to put their craft ahead of their tailors. And the most productive rank skills before degrees.

But once an industry starts creating wealth, the business grads and 3-piece suiters quickly rise to the top.

Woz worked in a little cubicle at Hewlett-Packard. With their blessing he left with the seeds of the most munificent Apple tree ever. Now he is back to working in a cubicle. Are other seeds incubating? Will they stay in the same orchard?

Lawsuits

Another kind of action is drawn by the magnet of success: legal. Friends suing friends for more than they ever made. Visicorp suing Software Arts for $50 million: "You were too slow putting advanced Visicalc onto the IBM-PC." Software Arts suing Visicorp for $87 million: "You didn't promote Visicalc well enough."

United Computer Corporation (why buy when you can rent) being sued by MicroPro and others. Maybe some people only rent so they can make their own copies. In any case, UCC shouldn't remove the license agreements from the packages!

UCC has also earned some lawsuits over their advertising debts. They prepay the first month, and ask for 30-day terms to run until further notice. That was last April...we caught on in July.

Following Suit

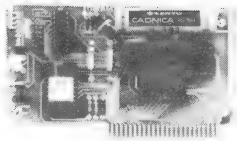
The whole world seems to be going IBM. Last year it was all CP/M. Next year it may be all AT&T. Remember back when everyone was copying Apple?

Businessmen buy those computers having the most on-going software and hardware development. Developers, programmers, cloners, and other entrepreneurs gather around systems businessmen are buying. The boys go where the girls are, which is where the boys are, which is where the girls are.... Being popular is so popular!

All of which slows down innovation in the marketplace. Not that innovation is all good and popularity is all bad. One secret of success is to stay the same long enough. Apple

APPLIED ENGINEERING
THE BEST PERIPHERALS FOR THE BEST COMPUTER

The TIMEMASTER
Finally a clock that does it ALL!



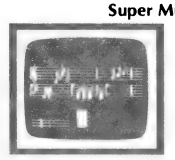
- Designed in 1983 using I.C. technologies that simply did not exist when most other Apple clocks were designed.
- Just plug it in and your programs can read the year, month, date, day, and time to 1 millisecond! The only clock with both year and ms.
- Powerful 2K ROM driver — No clock could be easier to use.
- Full emulation of most other clocks, including Mountain Hardware's Appleclock (but you'll like the TIMEMASTER mode better).
- Basic, Machine Code, CP/M and Pascal software on 2 disks!
- Eight software controlled interrupts so you can execute two programs at the same time. (Many examples are included)
- On board timer lets you time any interval up to 48 days long down to the nearest millisecond.

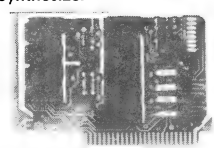
The TIMEMASTER includes 2 disks with some really fantastic time oriented programs (over 25) plus a DOS dater so it will automatically add the date when disk files are created or modified. This disk is over a $200.00 value alone — we give the software others sell. All software packages for business, data base management and communications are made to read the TIMEMASTER.

If you want the most powerful and the easiest to use clock for your Apple, you want a TIMEMASTER.

PRICE $129.00

Super Music Synthesizer

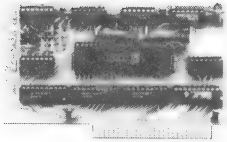




- Complete 16 voice music synthesizer on one card. Just plug it into your Apple, connect the audio cable (supplied) to your stereo, boot the disk supplied and you are ready to input and play songs.
- It's easy to program music with our compose software. You will start right away at inputting your favorite songs. The Hi-Res screen shows what you have entered in standard sheet music format.
- Now with new improved software for the easiest and fastest music input system available anywhere.
- We give you lots of software. In addition to Compose and Play programs, 2 disks are filled with over 30 songs ready to play.
- Easy to program in Basic to generate complex sound effects. Now your games can have explosions, phaser zaps, train whistles, death cries. You name it, this card can do it.
- Four white noise generators which are great for sound effects.
- Plays music in true stereo as well as true discrete quadraphonic.
- Full control of attack, volume, decay, sustain and release.
- Will play songs written for ALF synthesizer (ALF software will not take advantage of all the features of this board. Their software sounds the same in our synthesizer.)
- Automatic shutoff on power-up or if reset is pushed.
- Many many more features.

PRICE $159.00

Z-80 PLUS!



- TOTALLY compatible with ALL CP/M software.
- The only Z-80 card with a special 2K "CP/M detector" chip.
- Fully compatible with microsoft disks (no pre-boot required).
- All new 1983 design incorporates the latest in I.C. technologies.
- Red "CP/M WORKING" LED indicator, the Z-80 Plus does not interfere with non-CP/M programs.
- An on-card PROM eliminates many I.C.'s for a cooler, less power consuming board. (We use the Z-80A at a fast 4MHZ)
- Does EVERYTHING the other Z-80 boards do, plus Z-80 interrupts.

Don't confuse the Z-80 Plus with crude copies of the microsoft card. The Z-80 Plus employs a much more sophisticated and reliable design. With the Z-80 Plus you can access the largest body of software in existence. Two computers in one and the advantages of both, all at an unbelievably low price.

PRICE $139.00

COMING SOON: The Z-80 Plus for the Apple III

Viewmaster 80

There used to be about a dozen 80 column cards for the Apple, now there's only ONE.

- TOTALLY Videx Compatible
- 80 characters by 24 lines, with a sharp 7x9 dot matrix
- On-board 40/80 soft video switch with manual 40 column override
- Fully compatible with ALL Apple languages and software — there are NO exceptions
- Low power consumption through the use of CMOS devices
- All connections on the card are made with standard video connectors, no cables are soldered to the board
- All new 1983 design (using a new Microprocessor based C.R.T. controller)

JUST COMPARE!

	PRICE	REVERSE VIDEO SOFTWARE	SOFTWARE SCROLLING	PASCAL COMPATIBLE	CP/M/80 SOFTWARE	GRAPHICS CAPABILITY	80 COLUMN CURSOR	INTERNAL STRAP OPTIONS
VIEWMASTER	169	YES	YES	YES	YES	YES	YES	YES
SUPERTERM	375	NO	YES	NO	NO	NO	YES	YES
WIZARD80	245	NO	NO	YES	YES	NO	YES	YES
VISION80	375	YES	YES	YES	YES	NO	NO	NO
OMNIVISION	295	NO	YES	NO	NO	NO	YES	YES
VIEWMAX80	219	YES	YES	YES	YES	NO	NO	YES
SMARTERM	360	YES	YES	YES	NO	NO	YES	NO
VIDEOTERM	345	NO	NO	NO	YES	YES	NO	YES

The VIEWMASTER 80 works with all 80 column applications including CP/M, Pascal, WordStar, Format II, Easywriter, Apple Writer II, Viscalc, and many others. The VIEWMASTER 80 is THE MOST compatible 80 column card you can buy at ANY price!

PRICE $169.00

MemoryMaster IIe 128K RAM Card

- Expands your Apple IIe to 192K memory
- Provides an 80 column text display
- Compatible with all Apple IIe 80 column and extended 80 column card software (Same physical size as Apple's 64K card)
- Available in 64K and 128K configurations
- Bank select LED's for each 64K bank
- Permits your IIe to use the new double high resolution graphics
- Automatically expands Visicalc to 95K storage in 80 columns! The 64K configuration is all that's needed, 128K can take you even higher.
- Complete documentation included, we show you how to use all 128K.

If you already have Apple's 64K card, just order the MEMORYMASTER with 64K and use the 64K from your old board to give you a full 128K. (The board is fully socketed so you simply plug in more chips.)

MemoryMaster with 128K	**$249**
Upgradeable MemoryMaster with 64K	**$169**
Non-Upgradeable MemoryMaster with 64K	**$149**

Our boards are far superior to most of the consumer electronics made today. All I.C.'s are in high quality sockets with mil-spec. components used throughout. P.C. boards are glass-epoxy with gold contacts. Made in America to be the best in the world. All products work in APPLE IIe, II, II+ and Franklin (except MemoryMaster). Applied Engineering also manufactures a full line of data acquisition and control products for the Apple; A/D converters and digital I/O cards, etc. Please call for more information. All our products are fully tested with complete documentation and available for immediate delivery. All products are guaranteed with a no hassle **THREE YEAR WARRANTY.**

All Orders Shipped Same Day. Texas Residents Add 5% Sales Tax. Add $10.00 If Outside U.S.A. Dealer Inquiries Welcome.

Send Check or Money Order to:
APPLIED ENGINEERING
P.O. Box 470301
Dallas, TX 75247

Call (214) 492-2027
7a.m. to 11p.m. 7 days a week
MasterCard, Visa & C.O.D. Welcome

II/Plus/e has presented a stable yet growing environment for developers...contrast with Commodore/OSI/Radio Shack and their strings of mutually incompatible environments.

But innovators brought us the computer. And the supercomputer. And the minicomputer. And the microcomputer. And the Apple. And the....

More on the new 6502.....Bob Sander-Cederlof

I talked for about 15 minutes this morning (Dec 16th) with Bill Mensch. Bill used to work at Motorola, and was involved in the design of the original 6800 family there. Chuck Peddle joined the group, and noticed opportunities others were overlooking. Chuck and Bill decided to move to Pennsylvania, and with a few friends founded MOS Technology. They designed and built the 6501 microprocessor, but someone said it looked too much like the 6800 for comfort. Then came the 6502, leading to multiple millions of video games and personal computers. Bill is now at his own design company (Western Design Systems).

Bill told me he designed all the various CMOS versions of the 6502. Now he has designed the 65802 and 65816, CMOS versions with 16-bit registers and 16-megabyte address space. And he is currently working on a 32-bit version!

You probably read about these new versions on page 64 of the December Softalk, or in recent issues of Infoworld or Electronic Design. Elsewhere in Softalk you might also have noticed a box summarizing comments by Woz about plans for a new enhanced Apple //e with 16-megabyte capability. There are probably still other manufacturers out there with boxes and sockets just waiting on the first of Bill's new chips!

I just wish I could convey on paper how excited I am about this new chip! To me, it is as revolutionary as the original microprocessors were in their day. I predict that the 65816 and its successors will prove to be more powerful than the 68008: you will be able to write more compact code that runs faster, and build boards for less money that use less electricity.

With Bill's permission I am re-printing parts of his data sheet. You can get the complete package by calling (602) 962-4545 or writing to Western Design Center, 2166 E. Brown Rd, Mesa, Arizona 85203.

W65SC816

OXI-CMOS W65SC8XX and W65SC9XX 16-Bit Microprocessor Family

Features

- Advanced CMOS design for low power consumption and increased noise immunity
- Single +5V power supply
- Emulation mode allows complete hardware and software compatibility with NMOS 6502 code
- 24-bit address bus allows access to 16 MBytes of memory space
- Full 16-bit ALU, Accumulator, Stack Pointer, and Index Registers
- Valid Data Address (VDA) and Valid Program Address (VPA) output allows dual cache and cycle steal DMA implementation
- Vector Pull ($\overline{VP}$) output indicates when interrupt vectors are being addressed. May be used to implement vectored interrupt design
- Abort ($\overline{ABORT}$) input and associated vector supports interrupting any instruction without modifying memory or registers
- Separate program and data bank registers allow program segmentation
- New Direct Register allows "zero page" addressing anywhere in first 64K bytes
- 24 addressing modes—13 original 6502 modes, plus 11 new addressing modes
- New Wait for Interrupt (WAI) and Stop the Clock (STP) instructions further reduce power consumption, decrease interrupt latency and allows synchronization with external events
- New Co-Processor instruction (COP) with associated vector supports co-processor configurations, i.e., floating point processors
- New block move ability

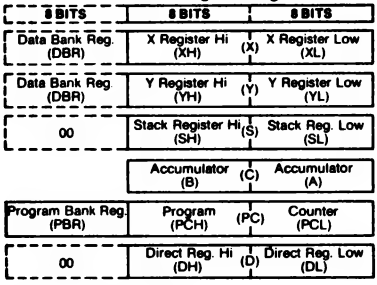
General Description

WDC's W65SC802 and W65SC816 are OXI-CMOS 16-bit microprocessors featuring total software compatibility with their 8-bit NMOS and CMOS 6500-series predecessors. The W65SC802 is pin-to-pin compatible with 8-bit devices currently available, while the W65SC816 extends addressing to a full 16 megabytes. These devices offer the many advantages of WDC's OXI-CMOS technology, including increased noise immunity, higher reliability, and greatly reduced power requirements. A software switch determines whether the processor is in the 8-bit "emulation" mode, or in the full 16-bit mode, thus allowing existing systems to use the expanded features.

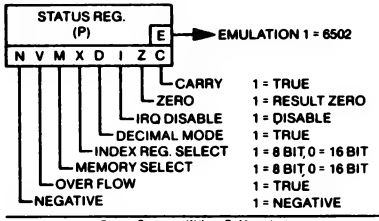
As shown in the processor programming model, the Accumulator, ALU, X and Y Index registers, and Stack Pointer register have all been extended to 16 bits. A new 16-bit Direct Page register augments the Direct Page addressing mode (formerly Zero Page addressing). Separate Program Bank and Data Bank registers allow 24-bit memory addressing.

Four new signals provide the system designer with many options. The $\overline{ABORT}$ input can interrupt the currently executing instruction without modifying internal registers. Valid Data Address (VDA) and Valid Program Address (VPA) outputs facilitate dual cache memory by indicating whether a data segment or program segment is accessed. Modifying a vector is made easy by monitoring the Vector Pull ($\overline{VP}$) output.

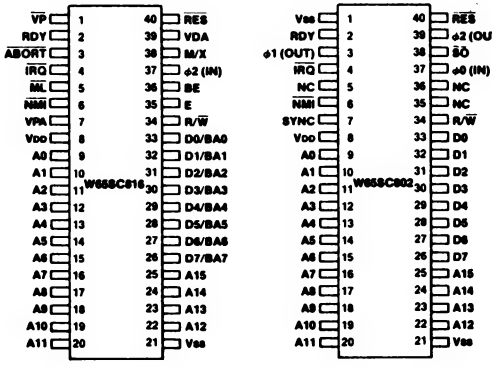
W65SC816 Processor Programming Model



Status Register Coding



Pin Configuration



Design Engineer: William D. Mensch, Jr.

THE WESTERN DESIGN CENTER, INC.
2166 East Brown Road • Mesa, Arizona 85203 • 602 962 4545

Advance Information Data Sheet:
This is advanced information and specifications are subject to change without notice.

Table 1. W65SC816 Compatibility Issues

- E = Emulation Bit which defines 6502 emulation mode
- XCE instruction exchanges carry bit C and emulation bit E

	W65SC816	W65SC02	NMOS 6502
1. S (Stack)	Always page 1 (E = 1) 16 bits when (E = 0).	Always page 1	Always page 1
2. X (X Index Register)	Indexed page zero always in page 0 (E = 1). Cross page (E = 0).	Always page 0	Always page 0
3. Y (Y Index Register)	Indexed page zero always in page 0 (E = 1). Cross page (E = 0).	Always page 0	Always page 0
4. A (Accumulator)	Same	Same	Same
5. P (Flag Registor)	N, V, and Z flags valid in decimal mode. (D not modified after reset or Interrupt E = 1). (D = 0 after Interrupt E = 0).	N, V, and Z flags valid in decimal mode. D = 0 after reset and Interrupt.	N, V, and Z flags invalid in decimal mode. D = unknown after reset. D not modified after Interrupt.
6. Timing			
A. ABS, X ASL, DEC, INC, LSR, ROL, POR With No Page Crossing	7 cycles	6 cycles	7 cycles
B. Jump Indirect Operand = XXFF	5 cycles	6 cycles	5 cycles and invalid page crossing
C. Branch Across Page	4 cycles (E = 1) 3 cycles (E = 0)	4 cycles	4 cycles
D. Decimal Mode	No additional cycle	Add 1 cycle	No additional cycles
7. BRK Vector	00FFFE,F (E = 1) BRK bit = 0 on stack if $\overline{IRQ}$, $\overline{NMI}$, $\overline{ABORT}$. 00FFE6, 7 (E = 0) X = X on Stack always.	FFFE,F BRK bit = 0 on stack if $\overline{IRQ}$, $\overline{NMI}$.	FFFE,F BRK bit = 0 on stack if $\overline{IRQ}$, $\overline{NMI}$.
8. Interrupt or Break Bank Address	PBR not pushed (E = 1) RTI PBR not pulled (E = 1) PBR pushed (E = 0) RTI PBR pulled (E = 0)	Not available	Not available
9. Memory Lock ($\overline{ML}$)	$\overline{ML}$ = 0 during Read, Modify and Write cycles.	$\overline{ML}$ = 0 during Modify and Write.	Not available
10. Indexed Across Page Boundary	Extra read of last instruction fetch.	Extra read of last instruction fetch.	Extra read of invalid address.
11. RDY Pulled During Write Cycle	Ignored (E = 1). Processor stops (E = 0).	Processor stops	Ignored
12. R/$\overline{W}$ During Reset Stack Operation	Does not write to stack.	Writes to stack	Does not write to stack.
13. Unused OP Codes	One reserved Op Code specified as WDM will be used in future systems. The W65SC816 performs a no-operation.	No operation	Unknown and some "hang up" processor.
14. Bank Address Handling	PBR = 00 after Reset or Interrupts.	Not available	Not available
15. R/$\overline{W}$ During Read-Modify-Write Instructions	E = 1, R/$\overline{W}$ = 0 during Modify and Write cycles. E = 0, R/$\overline{W}$ = 0 only during Write cycle.	R/$\overline{W}$ = 0 only during Write cycle	R/$\overline{W}$ = 0 during Modify and Write cycles.
16. SYNC (Metal Option)	W65SC802 VPA = SYNC. W65SC816 VPA = VPA Always.	SYNC Always	SYNC Always

---------- APPLE SOFTWARE ----------

NEW!!! FONT DOWNLOADER & EDITOR ($39.00)

Turn your printer into a custom typesetter. Downloaded characters remain active while printer is powered. Can be used with every word processor capable of sending ESC and control codes to the printer. Switch back and forth easily between standard and custom fonts. All special printer functions (like expanded, compressed, emphasized, underlined, etc.) apply to custom fonts. Full HIRES screen editor lets you create your own custom characters and special graphics symbols. Compatible with many 'dumb' & 'smart' printer I/F cards. User driver option provided. Specify printer: Apple Dot Matrix Printer, C.Itoh 8510A (Prowriter), Epson FX-80/100 or OkiData 92/93.

DISASM 2.2e - AN INTELLIGENT DISASSEMBLER ($30.00)

Investigate the inner workings of machine language programs. DISASM converts 6502 machine code into meaningful, symbolic source. Creates a standard DOS 3.3 text file which is directly compatible with DOS ToolKit, LISA and S-C (4.0 and MACRO) assemblers. Handles data tables, displaced object code & even lets you substitute your own meaningful labels. (100 commonly used Monitor & Pg Zero pg names included.) An address-based cross reference table provides further insight into the inner workings of machine language programs. DISASM is an invaluable machine language learning aid to both the novice & expert alike. SOURCE code: $60.00

S-C ASSEMBLER (Ver4.0 only) SUPPORT UTILITY PACKAGE ($30.00)

SC.XREF - Generates a GLOBAL LABEL Cross Reference Table for complete documentation of source listings. Formatting control accomodates all printer widths for best hardcopy outputs. # SC.GSR - Global Search and Replace eliminates tedious manual renaming of labels. Search all or part of source. Optional prompting for user verification. # SC.TAB - Tabulates source files into neat, readable form. SOURCE code: $40.00

---------- HARDWARE/FIRMWARE ----------

THE 'PERFORMER' CARD ($39.00)

Plugs into any Apple slot to convert your 'dumb' centronics-type printer I/F card into a 'smart' one. Command menu provides easy access to printer fonts. Eliminates need to remember complicated ESC codes and key them in to setup printer. Added features include perforation skip, auto page numbering with date & title. Also includes large HIRES graphics screen dump in normal or inverse plus full page TEXT screen dump. Specify printer: Epson MX-80 with Graftrax-80, MX-100, MX-80/100 with GraftraxPlus, NEC 8092JA, C.Itoh 8510 (Prowriter), OkiData 82A/83A with Okigraph & OpkiData 92/93. Oki bonus: print EMPHASIZED & DOUBLE STRIKE fonts! SOURCE code: $30.00

FIRMWARE FOR APPLE-CAT: The 'MIRROR' ROM ($25.00)

Communications ROM plugs directly into Novation's Apple-Cat Modem card. Three basic modes: Dumb Terminal, Remote Console & Programmable Modem. Added features include: selectable pulse or tone dialing, true dialtone detection, audible ring detect, ring-back option and built-in printer buffer. Supports most 80-column displays and the 1-wire shift key mod. Uses a superset of Apple's Comm card and Micromodem II commands. A-C hardware differences prevent 100% compatibility with Comm card. SOURCE code: $60.00

RAM/ROM DEVELOPMENT BOARD ($30.00)

Plugs into any Apple slot. Holds one user-supplied 2Kx8 memory chip. Use a 6116 type RAM chip for program development or just extra memory. Plug in a preprogrammed 2716 EPROM to keep your favorite routines 'on-line'. A versatile board with many uses! Maps into $Cn00-CnFF and $C800-CFFF memory space. Circuit diagram included.

NEW!!! SINGLE BOARD COMPUTER KIT ($20.00)

Kit includes etched PC board (with solder mask and plated thru holes) and assembly instructions. User provides 6502 CPU, 6116 2K RAM, 6821 dual 8-bit I/O and 2732 4K EPROM plus misc common parts. Originally designed as intelligent printer interface - easily adapted to many applications needing dedicated controller. (Assembled and tested: $119.00)

All assembly language SOURCE code is fully commented & provided in both S-C Assembler & standard TEXT formats on an Apple DOS 3.3 diskette. Specify your system configuration with order. Avoid a $3.00 postage and handling charge by enclosing full payment with order (MasterCard & VISA excluded). Ask about our products for the VIC-20 and Commodore 64!

R A K - W A R E
41 Ralph Road West Orange NJ 07052 (201) 325-1885

B. New W65SCXXX Instructions (13 Op Codes)

1. BRA — Branch Relative always
2. PLX — Pull X from Stack
3. PLY — Pull Y from Stack
4. PHX — Push X on Stack
5. PHY — Push Y on Stack
6. STZ — Store Zero in Memory (Direct; Direct, X; Abs; Abs, X)
7. TRB — Test and Reset Memory Bits Determined by Accumulator A (Direct and Absolute).
8. TSB — Test and Set Memory Bits Determined by Accumulator A (Direct and Absolute).

C. New W65SCXXX Addressing Modes (14 Op Codes)

2. BIT — Test Bits in Memory with Accumulator (Direct, X; Absolute, X; Immediate).
2. DEC — Decrement (Accumulator)
3. Group I — Instructions (Direct Indirect (8 Op Codes))
4. INC — Increment (Accumulator)
5. JMP — Jump to New Location (Absolute Indexed Indirect)

D. Group I Instructions with New Addressing Modes (48 Op Codes)

- Direct Indirect Long Indexed with Y (8 Op Codes)
- Direct Indirect Long (8 Op Codes)
- Absolute Long and Absolute Long Indexed with X (16 Op Codes)
- Stack Relative (8 Op Codes)
- Stack Relative Indirect Indexed Y (8 Op Codes)

1. ADC — Add Memory to Accumulator with Carry
2. AND — "AND" Memory with Accumulator
3. CMP — Compare Memory and Accumulator
4. EOR — "Exclusive-or" Memory with Accumulator
5. LDA — Load Accumulator with Memory
6. ORA — "Or" Memory with Accumulator
7. SBC — Subtract Memory from Accumulator with Borrow
8. STA — Store Accumulator in Memory

E. New Push and Pull Instructions (7 Op Codes)

1. PEA — Push Effective Absolute Address or Immediate Data Word on Stack
2. PEI — Push Effective Indirect Address or Direct Data Word on Stack
3. PER — Push Effective Program Counter Relative Indirect Address or Program Counter Relative Data Word on Stack
4. PLB — Pull Data Bank Register from Stack
5. PLD — Pull Direct Register from Stack
6. PHB — Push Data Bank Register on Stack
7. PHD — Push Direct Register on Stack
8. PHK — Push Program Bank Register on stack

F. Status Register Instructions (2 Op Codes)

1. REP — Reset Status Bits Defined by Immediate Byte 1 = Reset
   0 = Do not change
2. SEP — Set Status Bits Defined by Immediate Byte 1 = Set
   0 = Do not change

G. New Register Transfer Instructions (8 Op Codes)

1. TCD — Transfer C Accumulator to Direct Register D
2. TDC — Transfer Direct Register D to C Accumulator
3. TCS — Transfer C Accumulator to Stack Register
4. TSC — Transfer Stack Register to Accumulator C
5. TXY — Transfer X to Y
6. TYX — Transfer Y to X
7. XBA — Exchange B and A
8. SCE — Exchange Carry Bit C with Emulation Bit E.

H. New Branch, Jump and Return Instructions (6 Op Codes)

1. BRL — Branch Relative Long Always (16 Bit Relative—32768 to + 32767) (Addressing Mode)
2. JML — Jump Indirect Long
3. JMP — Jump Absolute Long
4. JSL — Jump to Subroutine Long (Uses RTL for Return)
5. JSR — Jump to Subroutine (Indexed Indirect)
6. RTL — Return from Subroutine Long

I. New Block Move Instructions (2 Op Codes)

1. MVN — Move Block from Source (X Addressed) to Destination (Y Addressed). Block Length Defined by C. X, Y are Incremented.
2. MVP — Move Block from Source (X Addressed) to Destination (Y Addressed). Block Length Defined by C. X, Y are Decremented.

J. New Co-Processor Operations (1 Op Code)

1. COP — Co-Processor Instruction with Associated COP Vector and ABORT Input Supports Co-Processing Function i.e., Floating Point Processors, etc.

K. New System Control Instructions (3 Op Codes)

1. STP — Stop-the-clock Instruction Stops the Oscillator Input (or 02 Input) During 02 = 1. This Mode Is Released When RES Goes to a Zero. System Initialization May Be Desired; However, if After RESET One Performed an RTI, Program Execution Begins With the Instruction Following the STP Op Code in Program Sequence.
2. WAI — Wait for Interrupt Pulls RDY Low and Is Cleared by IRQ or NMI Active Input.
3. WDM — There is One Reserved Op Code Defined as WDM Which Will Be Used For Future Systems. The W65SC816 Performs a No-Operation.

Addressing Modes

Twenty-four addressing modes are available to the user of the W65SC816 family of microprocessors. The addressing modes are described in the following paragraphs.

1. Immediate Addressing [imm]
With immediate addressing the operand is contained in the second byte (second and third byte for 16 bit data) of the instruction.

2, 3. Absolute and Absolute Long Addressing [a], [al]
For absolute addressing the second byte of the instruction specifies the eight low order bits of the effective address while the third byte specifies the eight high order bits. For absolute long addressing the fourth byte specifies the bank address. The full 16.7 megabyte address space is addressed in the long mode. In the short mode the bank address is specified by the data bank register.

4. Direct Addressing [d]
Direct addressing allows for shorter code and execution times by only fetching a second byte of instruction. The second byte is added to the direct register (D) value. When the direct register low (DL) is zero fastest execution occurs. The bank address is always zero.

5. Accumulator Addressing [acc]
This form of addressing is represented with a one byte instruction and performs an operation on the accumulator(s).

6. Implied Addressing [imp]
In the implied addressing mode the address of the operand is implicitly stated in the operation code of the instruction.

7, 8. Direct Indirect Indexed and Direct Indirect Indexed Long Addressing [(d), y], [(dl), y]
This form of addressing is usually referred to as Indirect, Y. The second byte of the instruction is added to the direct register and points to a memory location in bank zero. The contents of this memory location and the byte following (the next byte is the bank address for the long mode) are added to the Y index register with the result being the effective address. For the short mode the bank address is specified by the data bank register. Note that when DL equals zero execution is fastest.

9. Direct Indexed Indirect Addressing [(d,x)]
With direct indexed indirect addressing (usually referred to as Indirect X) the second byte of the instruction is added to the contents of the direct register and then adding the X register value. The result of these additions points to a memory location on bank zero whose contents is the low order byte of the effective address with the byte following the high byte of the effective address. The bank address of the effective address is specified by the data bank register.

10, 11. Direct Indexed with X and Direct Indexed with Y Addressing [d,x], [d,y]
Direct indexed with X usually referred to as Direct, X and direct indexed with Y usually referred to as Direct, Y are two byte instructions. The second byte is added to the direct register (D) and this result is added to the appropriate index register. The bank address is always zero. Execution is fastest when the low byte of the direct register (DL) is zero.

12, 13, 14. Absolute Indexed with X, Absolute Indexed Long with X, and Absolute Indexed with Y Addressing [a,x], [al,x], [a,y]
Absolute indexed addressing is used in conjunction with the X and Y index registers and is referred to as Absolute, X Absolute Long, X and Absolute, Y. The effective address is formed by adding the contents of the X or Y register to the second and third bytes of the instructions. The bank address is specified by the data bank register except in the long mode the fourth byte specifies the bank address.

15, 16. Program Counter Relative and Program Counter Relative Long Addressing [r], [rl]
Program counter relative addressing, usually referred to as relative and relative long addressing is used only with the branch instructions. The second byte is added to the program counter which for relative creates a +128 or -127 byte offset. The second and third bytes are added to the program counter to create +32768 or -32767 byte offset for the branch always long operation.

17. Absolute Indirect Addressing (Jump Instruction Only) [(a)]
The second and third bytes of the instruction contains the low and high order address bytes of a memory location located in bank zero. This memory location and the byte following contain the effective address which is loaded into the program counter. The destination bank address is specified by the program bank register except for the JML instruction the third byte fetched is the destination bank address.

18, 19. Direct Indirect and Direct Indirect Long Addressing [(d)], [(dl)]
In this form of addressing the second byte of the instruction is added to the direct register and the result points to a memory location in bank zero. The contents of this location and the following location (the next location is the bank address for the long mode) is the effective address. The bank address is specified by the data bank register for the direct indirect mode.

20. Absolute Indexed Indirect Addressing (Jump and Jump to Subroutine) [(a,x)]
With absolute indexed indirect addressing the second and third bytes of the instruction are added to the X index register contents. The result points to the low and (byte following) high order bytes which are loaded into the program counter. The bank address is specified by the program bank register.

21. Stack Addressing [s]
This addressing mode uses the stack register to address memory locations. The instructions which use the stack addressing include push, pull, interrupts, jump to subroutine, return from interrupt and return from subroutine. The bank address is always zero. Vectors are always pulled from bank 00. (See Compatibility Issues for 6502 Emulation)

22. Stack Relative Addressing [sr]
With stack relative addressing the second byte of the instruction is added to the stack register value. This effective address points to a data memory location on the stack. For 16 bit data the next location on the stack is the high byte of data. This addressing mode, in conjunction with using the push instructions, may be used to pass data to subroutines using the stack. The new TSC and TCS instructions provide fast stack modification. The direct register can be used for user stack functions. The bank register is always zero.

23. Stack Relative Indirect Indexed Addressing [(sr),y]
With stack relative indirect indexed with Y the second byte of the instruction is added to the stack register value. The address formed by this addition points to the low byte (the next location contains the high byte) of an indirect address. The Y register is added to this address to form the effective data address. This addressing mode, in conjunction with using the push effective address (PEA, PEI, PER) instructions, may be used to pass data addresses to subroutines using the stack. The new TSC and TCS instructions provide fast stack register modification. The direct register can be used for user stack functions. The data bank register is the bank address for the effective address.

24. Block Move Addressing [xyc]
This addressing mode is used for multiple byte moves forward (MVP) or backward (MVN). These three byte instructions use the X register for the source address, the Y register for the destination address and the C accumulator contains the number of bytes to be moved. The destination bank address is the second byte of the instruction with the source bank specified by the third byte. The data bank register is loaded with the destination bank value (second byte of the instruction).

Table 4. W65SC816 Microprocessor Op Code Matrix

MSD	LSD 0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	MSD
0	BRK s 2 8	ORA(d,x) 2 6	COP s 2 *8	ORA sr 2 *4	TSB d 2 •5	ORA d 2 3	ASL d 2 5	ORA(dl) 2 *6	PHP s 1 3	ORA imm 2 2	ASL acc 1 2	PHD s 1 *4	TSB a 3 •6	ORA a 3 4	ASL a 3 6	ORA al 4 *5	0
1	BPL r 2 2	ORA(d),y 2 5	ORA (d) 2 •5	ORA(sr),y 2 *7	TRB d 2 •5	ORA d,x 2 4	ASL d,x 2 6	ORA(dl),y 2 *6	CLC imp 1 2	ORA a,y 3 4	INC acc 1 •2	TCS imp 1 *2	TRB a 3 •6	ORA a,x 3 4	ASL a,x 3 7	ORA al,x 4 *5	1
2	JSR a 3 6	AND(d,x) 2 6	JSL al 4 *8	AND sr 2 *4	BIT d 2 3	AND d 2 3	ROL d 2 5	AND(dl) 2 *6	PLP s 1 4	AND imm 2 2	ROL acc 1 2	PLD s 1 *5	BIT a 3 4	AND a 3 4	ROL a 3 6	AND al 4 *5	2
3	BMI r 2 2	AND(d),y 2 5	AND(d) 2 •5	AND(sr),y 2 *7	BIT d,x 2 •4	AND d,x 2 4	ROL d,x 2 6	AND(dl),y 2 *6	SEC imp 1 2	AND a,y 3 4	DEC acc 1 •2	TSC imp 1 *2	BIT a,x 3 •4	AND a,x 3 4	ROL a,x 3 7	AND al,x 4 *5	3
4	RTI s 1 7	EOR(d,x) 2 6	WDM RESERVED	EOR sr 2 *4	MVP xyc 3 *7	EOR d 2 3	LSR d 2 5	EOR(dl) 2 *6	PHA s 1 3	EOR imm 2 2	LSR acc 1 2	PHK s 1 *3	JMP a 3 3	EOR a 3 4	LSR a 3 6	EOR al 4 *5	4
5	BVC r 2 2	EOR(d),y 2 5	EOR (d) 2 •5	EOR(sr),y 2 *7	MVN xyc 3 *7	EOR d,x 2 4	LSR d,x 2 6	EOR(dl),y 2 *6	CLI imp 1 2	EOR a,y 3 4	PHY s 1 •3	TCD imp 1 *2	JMP al 3 *4	EOR a,x 3 4	LSR a,x 3 7	EOR al,x 4 *5	5
6	RTS s 1 6	ADC(d,x) 2 6	PER s 3 *6	ADC sr 2 *4	STZ d 2 •3	ADC d 2 3	ROR d 2 5	ADC(dl) 2 *6	PLA s 1 4	ADC imm 2 2	ROR acc 1 2	RTL s 1 *6	JMP (a) 3 5	ADC a 3 4	ROR a 3 6	ADC al 4 *5	6
7	BVS r 2 2	ADC(d),y 2 5	ADC(d) 2 •5	ADC(sr),y 2 *7	STZ d,x 2 •4	ADC d,x 2 4	ROR d,x 2 6	ADC(dl),y 2 *6	SEI imp 1 2	ADC a,y 3 4	PLY s 1 •4	TDC imp 1 *2	JMP(a,x) 3 •6	ADC a,x 3 4	ROR a,x 3 7	APC al,x 4 *5	7
8	BRA r 2 •2	STA(d,x) 2 6	BRL rl 3 *3	STA sr 2 *4	STY d 2 3	STA d 2 3	STX d 2 3	STA(dl) 2 *6	DEY imp 1 2	BIT imm 2 •2	TXA imm 1 2	PHB s 1 *3	STY a 3 4	STA a 3 4	STX a 3 6	STA al 4 *5	8
9	BCC r 2 2	STA(d),y 2 6	STA (d) 2 •5	STA(sr),y 2 *7	STY d,x 2 4	STA d,x 2 4	STX d,y 2 4	STA(dl),y 2 *6	TYA imp 1 2	STA a,y 3 5	TXS imp 1 2	TXY imp 1 *2	STZ a 3 •4	STA a,x 3 5	STZ a,x 3 •5	STA al,x 4 *5	9
A	LDY imm 2 2	LDA(d,x) 2 6	LDX imm 2 2	LDA sr 2 *4	LDY d 2 3	LDA d 2 3	LDX d 2 3	LDA(dl) 2 *6	TAY imp 1 2	LDA imm 2 2	TAX imp 1 2	PLB s 1 *4	LDY a 3 4	LDA a 3 4	LDX a 3 4	LDA al 4 *5	A
B	BCS r 2 2	LDA(d),y 2 5	LDA(d) 2 •5	LDA(sr),y 2 *7	LDY d,x 2 4	LDA d,x 2 4	LDX d,y 2 4	LDA(dl)y 2 *6	CLV imp 1 2	LDA a,y 3 4	TSX imp 1 2	TYX imp 1 *2	LDY a,x 3 4	LDA a,x 3 4	LDX a,y 3 4	LDA al,x 4 *5	B
C	CPY imm 2 2	CMP(d,x) 2 6	REP imm 2 *3	CMP sr 2 *4	CPY d 2 3	CMP d 2 3	DEC d 2 5	CMP(dl) 2 *6	INY imp 1 2	CMP imm 2 2	DEX imm 1 2	WAI imp 1 *3	CPY a 3 4	CMP a 3 4	DEC a 3 6	CMP al 4 *5	C
D	BNE r 2 2	CMP(d),y 2 5	CMP (d) 2 •5	CMP(sr),y 2 *7	PEI s 2 *6	CMP d,x 2 4	DEC d,x 2 6	CMP(dl),y 2 *6	CLD imp 1 2	CMP a,y 3 4	PHX s 1 •3	STP imp 1 *3	JML (a) 3 *6	CMP a,x 3 4	DEC a,x 3 7	CMP al,x 4 *5	D
E	CPX imm 2 2	SBC(d,x) 2 6	SEP imm 2 *3	SBC sr 2 *4	LPX d 2 3	SBC d 2 3	INC d 2 5	SBC(dl) 2 6	INX imp 1 2	SBC imm 2 2	NOP imp 1 2	XBA imp 1 *3	CPX a 3 4	SBC a 3 4	INC a 3 6	SBC al 4 *5	E
F	BEQ r 2 2	SBC(d),y 2 5	SBC (d) 2 •5	SBC(sr),y 2 *7	PEA s 3 *5	SBC d,x 2 4	INC d,x 2 6	SBC(dl),y 2 *6	SED imp 1 2	SBC a,y 3 4	PLX s 1 •4	XCE imp 1 *2	JSR(a,x) 3 *6	SBC a,x 3 4	INC a,x 3 7	SBC al,x 4 *5	F

* New W65SC816 Op Codes
• W65SC02 Op Codes

INSTRUCTION MNEMONIC
(COMMENT)
BASE NO. BYTES
ADDRESSING MODE
BASE NO. CYCLES

68000 "Color Pattern".....Bob Urschel
Valparaiso, Indiana

I have had my QWERTY Q68 board for about 2 weeks now. In my opinion this seems to be an excellent product and also a very inexpensive way to learn about the MC68000 MPU.

As an exercise I rewrote an Integer BASIC program called "ROD'S COLOR PATTERN" found in my red Apple II Reference Manual (1978). I am sending you two versions of my program. the first version does the calculation for the LORES screen base addresses. The second version looks up the base addresses in a table, consequently running slightly faster than the first version. The second version runs (as close as I can tell) about 50 times faster than the Integer BASIC program.

[We're printing only the first version, since the GBASCALC routine is more interesting than a table lookup. QD 14 will include both version... Bill]

After the 68000 source code has been assembled, I BRUN a very short 6502 program which consists of the following code:

```
              .OR $1080
              JSR $30B     TURN ON THE Q68 BOARD
HERE          JMP HERE     DON'T DO ANYTHING
```

This keeps the 6502 busy while the 68000 is doing all the work.

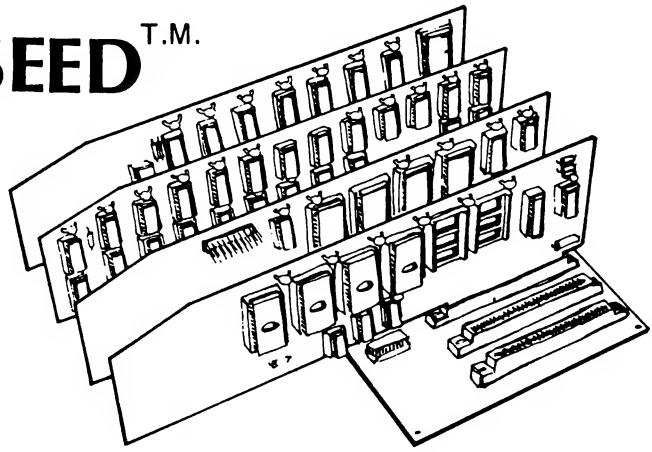
Here's a listing of ROD'S COLOR PATTERN in Integer BASIC:

```
10 GR
20 FOR W = 3 TO 50
30 FOR I = 1 TO 19
40 FOR J = 0 TO 19
50 K = I+J
60 COLOR = J*3 / (I+3) + I*W/12
70 PLOT I,K: PLOT K,I: PLOT 40-I,40-K: PLOT 40-K,40-I
80 PLOT K,40-I: PLOT 40-I,K: PLOT I,40-K: PLOT 40-K,I
90 NEXT J: NEXT I: NEXT W
100 GOTO 20
```

```
                    1000 *SAVE S.URSCHEL'S COLOR PATTERN
                    1010 *-------------------------------
                    1020 *   RODS COLOR PATTERN
                    1030 *   RE-WRITTEN BY BOB URSCHEL
                    1040 *   USING THE QWERTY Q68 MC68000 MPU
                    1045 *
                    1047         .OR     $1000
00001000- 207C 0000
00001004- 1100      1050         MOVE.L  #$1100,A0  MOVE PROGRAM TO FAST MEMORY
00001006- 227C 0001
0000100A- 8600      1060         MOVE.L  #$18600,A1
0000100C- 323C 010D 1070         MOVE    #END-START,D1
00001010- 12D8      1080 XFER    MOVE.B  (A0)+,(A1)+
00001012- 51C9 FFFC 1090         DBF     D1,XFER
00001016- 4EF9 0001
0000101A- 8600      1095         JMP     $18600
                    1100 *
```

```
                        1110 *--------------------------------
                        1120 *
                        1140         .OR     $18600
                        1150         .TA     $1100
                        1160 START
00018600- 4A39 0000
00018604- C050          1170         TST.B   $C050      >GR
00018606- 6100 00F0     1180         BSR     CLRSCR     CLEAR SCREEN
                        1190 *--------------------------------
                        1200 START.W
0001860A- 13FC 0003
0001860E- 0001 870C     1210         MOVE.B  #3,W       >FOR W = 3 TO 50
                        1220 START.I
00018612- 7E01          1230         MOVEQ   #1,D7      >FOR I = 1 TO 19
                        1240 START.J
00018614- 7600          1250         MOVEQ   #0,D3      >FOR J = 0 TO 19
00018616- 3C07          1260 SET.K   MOVE    D7,D6      >K = I + J
00018618- DC03          1270         ADD.B   D3,D6
                        1280 *--------------------------------
0001861A- 7000          1290         MOVEQ   #0,D0      >COLOR = J*3/(I+3)+I*W/12
0001861C- 3003          1300         MOVE    D3,D0
0001861E- C0FC 0003     1310         MULU    #3,D0      J*3
00018622- 7200          1320         MOVEQ   #0,D1
00018624- 3207          1330         MOVE    D7,D1
00018626- 5641          1340         ADDQ    #3,D1      I+3
00018628- 80C1          1350         DIVU    D1,D0      J*3/(I+3) --> D0
0001862A- 3207          1360         MOVE    D7,D1
0001862C- 7400          1370         MOVEQ   #0,D2
0001862E- 1439 0001
00018632- 870C          1380         MOVE.B  W,D2
00018634- C4C1          1390         MULU    D1,D2      I*W --> D2
00018636- 84FC 000C     1400         DIVU    #12,D2     D2 / 12
0001863A- D440          1410         ADD     D0,D2
0001863C- 0202 000F     1420         ANDI.B  #$F,D2
00018640- 13C2 0001
00018644- 870B          1430         MOVE.B  D2,COLOR   SET COLOR
                        1440 *
                        1450 *
                        1460 *  SUBTRACT I AND K FROM 40
                        1470 *
00018646- 7A28          1480         MOVEQ   #40,D5
00018648- 9A47          1490         SUB     D7,D5      D5 = 40 - I
0001864A- 7828          1500         MOVEQ   #40,D4
0001864C- 9846          1510         SUB     D6,D4      D4 = 40 - K
0001864E- 3007          1520         MOVE    D7,D0      >PLOT I,K
00018650- 3206          1530         MOVE    D6,D1
00018652- 6150          1540         BSR.S   PLOT
00018654- 3006          1550         MOVE    D6,D0      >PLOT K,I
00018656- 3207          1560         MOVE    D7,D1
00018658- 614A          1570         BSR.S   PLOT
0001865A- 3005          1580         MOVE    D5,D0      >PLOT 40-I,40-K
0001865C- 3204          1590         MOVE    D4,D1
0001865E- 6144          1600         BSR.S   PLOT
00018660- 3004          1610         MOVE    D4,D0      >PLOT 40-K,40-I
00018662- 3205          1620         MOVE    D5,D1
00018664- 613E          1630         BSR.S   PLOT
00018666- 3006          1640         MOVE    D6,D0      >PLOT K,40-I
00018668- 3205          1650         MOVE    D5,D1
0001866A- 6138          1660         BSR.S   PLOT
0001866C- 3005          1670         MOVE    D5,D0      >PLOT 40-I,K
0001866E- 3206          1680         MOVE    D6,D1
00018670- 6132          1690         BSR.S   PLOT
00018672- 3007          1700         MOVE    D7,D0      >PLOT I,40-K
00018674- 3204          1710         MOVE    D4,D1
00018676- 612C          1720         BSR.S   PLOT
00018678- 3004          1730         MOVE    D4,D0      >PLOT 40-K,I
0001867A- 3207          1740         MOVE    D7,D1
0001867C- 6126          1750         BSR.S   PLOT
0001867E- 5243          1760         ADDQ    #1,D3      >NEXT J
00018680- 0C43 0014     1770         CMPI    #20,D3
00018684- 6690          1780         BNE     SET.K
00018686- 5247          1790         ADDQ    #1,D7      >NEXT I
00018688- 0C47 0014     1800         CMPI    #20,D7
0001868C- 6686          1810         BNE     START.J
```

APPLESEED T.M.



Appleseed is a complete computer system. It is designed using the bus conventions established by Apple Computer for the Apple][. Appleseed is designed as an alternative to using a full Apple][computer system. The Appleseed product line includes more than a dozen items including CPU, RAM, EPROM, UART, UNIVERSAL Boards as well as a number of other compatible items. This ad will highlight the Mother board.

BX-DE-12 MOTHER BOARD

The BX-DE-12 Mother board is designed to be fully compatible with all of the Apple conventions. Ten card slots are provided. Seven of the slots are numbered in conformance with Apple standards. The additional three slots, lettered A, B and C, are used for boards which don't require a specific slot number. The CPU, RAM and EPROM boards are often placed in the slots A, B and C.

The main emphasis of the Appleseed system is illustrated by the Mother Board. The absolute minimum amount of circuitry is placed on the Mother Board; only the four ICs which are required for card slot selection are on the mother board. This approach helps in packaging (flexibility & smaller size), cost (buy only what you need) and repairability (isolate and fix problems through board substitution).

Appleseed products are made for O.E.M.s and serious industrial/scientific users. Send for literature on the full line of Appleseed products; and, watch here, each month, for additional items in the Appleseed line.

Appleseed products are not sold through computer stores.

Order direct from our plant in California.

Apple is a registered trademark of Apple Computer, Inc.

DOUGLAS ELECTRONICS
718 Marina Blvd., San Leandro, CA 94577 • (415) 483-8770

```
0001868E- 5239 0001
00018692- 870C       1820         ADDQ.B  #1,W        >NEXT W
00018694- 0C39 0033
00018698- 0001 870C  1830         CMPI.B  #51,W
0001869C- 6700 FF6C  1840         BEQ     START.W
000186A0- 6600 FF70  1850         BNE     START.I
                     1860 *
                     1870 *-------------------------------
                     1880 *       PLOT SUBROUTINE
                     1890 *
                     1900 *       A0 = SCREEN ADDRESS
                     1910 *       D0 = X-COORD
                     1920 *       D1 = Y-COORD
                     1930 *       D2 = WORK REGISTER
                     1940 *
000186A4- 3040       1950 PLOT    MOVE    D0,A0       SAVE X-COORD
000186A6- E209       1960         LSR.B   #1,D1       GET CARRY
000186A8- 40C0       1970         MOVE    SR,D0       SAVE ODD-EVEN STATUS
000186AA- 6130       1980         BSR.S   GBASCALC
000186AC- D0C1       1990         ADD     D1,A0       FINAL SCREEN ADDR
000186AE- 13FC 00F0
000186B2- 0001 870A  2000         MOVE.B  #$F0,MASK
000186B6- 1239 0001
000186BA- 870B       2010         MOVE.B  COLOR,D1
000186BC- 44C0       2020         MOVE    D0,CCR      ODD OR EVEN?
000186BE- 640A       2030         BCC.S   PLOT1       EVEN...
000186C0- 13FC 000F
000186C4- 0001 870A  2040         MOVE.B  #$F,MASK
000186C8- E909       2050         LSL.B   #4,D1       ROTATE COLOR
000186CA- 1410       2060 PLOT1   MOVE.B  (A0),D2     ORIGINAL BYTE
000186CC- C439 0001
000186D0- 870A       2070         AND.B   MASK,D2     MASK OUT OLD COLOR
000186D2- 8401       2080         OR.B    D1,D2        AND GET NEW COLOR
000186D4- 1082       2090         MOVE.B  D2,(A0)     PLOT TO SCREEN
000186D6- 7000       2100         MOVEQ   #0,D0       CLEAR OUT CCR
000186D8- 7200       2110         MOVEQ   #0,D1
000186DA- 4E75       2120         RTS
                     2130 *
                     2140 *       CALCULATE BASE ADDRESS
                     2150 *
                     2160 GBASCALC
000186DC- 3401       2170         MOVE    D1,D2       000DEFGH
000186DE- 0201 0018  2180         AND.B   #$18,D1     000DE000
000186E2- EB0A       2190         LSL.B   #5,D2       FGH00000
000186E4- 8202       2200         OR.B    D2,D1       FGHDE000
000186E6- 1401       2210         MOVE.B  D1,D2
000186E8- 0202 0018  2220         AND.B   #$18,D2     000DE000
000186EC- E40A       2230         LSR.B   #2,D2       00000DE0
000186EE- 8202       2240         OR.B    D2,D1       FGHDEDE0
000186F0- 0041 0100  2250         OR      #$100,D1  1FGHDEDE0
000186F4- E549       2260         LSL     #2,D1   1FGHDEDE000
000186F6- 4E75       2270         RTS
                     2280 *
                     2290 *       CLEAR LORES SCREEN
                     2300 *
000186F8- 4240       2310 CLRSCR  CLR     D0
000186FA- 323C 01FF  2320         MOVE    #511,D1    # OF WORDS TO MOVE MINUS 1
000186FE- 307C 0800  2330         MOVE    #$800,A0   ENDING SCREEN ADDR
00018702- 3100       2340 .1      MOVE    D0,-(A0)
00018704- 51C9 FFFC  2350         DBF     D1,.1
00018708- 4E75       2360         RTS
                     2370 *-------------------------------
                     2380 *       WORK AND STORAGE
                     2390 *
0001870A-            2400 MASK    .BS     1
0001870B-            2410 COLOR   .BS     1
0001870C-            2420 W       .BS     1
                     2430 *
                     2440 *-------------------------------
                     2450 END
                     2460         .OR     $800
00000800- 0001 8800  2470         .DA     $18800
00000804- 0000 1000  2480         .DA     $1000
```

"Understanding the Apple II", a Review.....Bob Sander-Cederlof

If you want the real inside scoop on the Apple II, you need "Understanding the Apple II". Following close on the heels of Gayler's "Apple II Circuit Description", this book is no second-place sequel.

"Understanding..." was written by Jim Sather, a former ITT technical representative, after many moons of trial-and-error, pick-and-shovel research into the inner sanctum of our favorite computer. Jim has a gift for clearly explaining how things work. My degree is a little rusty, or mildewed, or whatever, and hardware never was my long suit. But Jim makes it all make sense for me.

The process of "understanding" starts with a few full color diagrams and charts. In the back of the book there are two foldout full color charts of bus structure and chip layout. Surprisingly, you find color sprinkled throughout the book, along with many black & white illustrations, photos, tables, diagrams, etc.

Sather describes microcomputer fundamentals with specific applications to the Apple II. He carefully documents all the circuits on the motherboard, as well as the firmware and language cards, and Wozniak's patented disk controller.

The chapter on the disk drive and controller is especially thorough, devoting some 45 large pages, including many diagrams, to the exact workings of these devices. I have never seen a better explication of the Apple's unique disk controller.

There are especially useful discussions of address decoding, RAM/ROM addressing, and bus structure. Sather's readable style avoids much of the reference-book prose common to authoritative technical books.

Each chapter ends with some of nearly two dozen hardware & software projects, including reprogramming screen character sets, an NMI based single stepper, detecting and using television sync, modifying the firmware card so the F8 ROM can be switch-selected, and more.

"Understanding..." begins with a foreword by Steve Wozniak, and ends near an appendix describing a conversation with Woz about some of the original design decisions that made our Apples what they are today.

This would be a good text book in computer hardware fundamentals at high school level or above. Most of the courses I took in college (now over 25 years ago!) were rather abstract and difficult to relate to real applications. What better way to understand how computers work, how they can be modified and maintained, and how to design them, than to dissect a living breathing example like the Apple II!

Here's a quick look at the structure of "Understanding the Apple II":

Chapters
1 Overview
2 Bus Structure
3 Timing Generation and the Video Scanner
4 The 6502 Microprocessor
5 RAM
6 ROM
7 Address Decoding and I/O
8 Video Generation
9 Disk Controller
10 Maintenance and Care

Glossary of 7 pages, about 150 entries.

Appendices: references, trademarks, 6502 data sheets, program listings, logic circuits primer, number systems primer, apple ii revisional info, historical notes, conversation with Woz, how to remove the motherboard, list of figures and tables.

Schematics, Index

"Understanding the Apple II" describes the Apple II and Apple II Plus. Much of the book's information, especially the chapter on the disk controller, applies also to the Apple //e. "Understanding the Apple //e" is promised sometime in 1984.

Understanding the Apple II, Jim Sather. About 356 pages. Quality Software, $22.95 (Buy it from us for only $21 + shipping).

Locksmith 5.0 Reviewed.....Bob Sander-Cederlof

I received my copy of Locksmith 5.0 last week. I haven't tried any of the lock-busting capabilities, because I have no particular need for that. But there are other features which justify the price. The new manual has information on copy protection schemes which I think has never been published before. The new manual is 140 pages long! I remember the first edition came with a tiny 1/2 page summary of operation!

The other item I am in love with is the fast copy program for ordinary DOS 3.3 disks. In a 64K Apple with two disk drives on one controller, it will make a copy in only 19 seconds! And if you have a larger memory (32K beyond a 128K //e, or a II with a 128K card), it can make a complete copy in only 16 seconds. And if you want to make multiple copies of the same disk, and have large enough memory to hold an entire disk image, you can make additional copies in 8 seconds flat! These copies are without verify, but a verify pass only adds 7 more seconds.

I think this one feature is worth the $99.95 price tag, but there are many more reasons for owning a copy. If you own a previous version, they have an attractive upgrade policy. If not, we will send you a copy for $90 + shipping.

On-Line with Steve Wozniak

Steve Knouse sent a printout of the "on-line with Woz" session you may have heard about. Some intriguing Woz-words:

About ProDOS use of >64K memory: "Our enhanced //e family is headed toward 16M bytes in short time with a revolutionary 6502-based processor."

About software for extended RAM cards: "I promise an alternative solution soon (6 mo?) for direct addressing of 24-bit address."

About MAC: "...look at LISA. Then imagine slightly fewer resources and memory but advantage taken to make it faster and better with fewer resources (sound familiar //e world?). Mouse, no color, no slots, finest software (BASIC and Pascal are finest ever done too). MAC will use its own op-sys which was developed to handle the user interface of LISA more directly with better performance. Such good software has been written for MAC (128K bytes in ROM) that it will be transferred to LISA soon!" "Initially MAC won't displace the PC as a small business machine but is intended to be a more finished product for the bulk of the personal market -- assuming which peripherals and features they would want and supplying them at lower cost than if they have slots to make their own choices. Interesting." "I believe that MAC is the most revolutionary computer of all time -- not that what it does hasn't been done before, but that it hasn't been done at a price which will wind up with millions experiencing it." "The MAC unfortunately is so perfect that we didn't leave much room for hackers to do hardware 'for themselves' or 'their own way' -- we feel there were no alternatives. The philosophy on software is different -- open, access the hardware at various levels."

About larger ProFILEs: "...yes, plans for larger ProFILEs. Pretty the minimal hard disk for small business has grown to 10MB, soon 20."

About the Apple: "The Apple II was not built to be a product for sale. It looked like the best thing available in 1976. The first computer ever (low cost) with color, hi-res, Basic in ROM, plastic case, switching power supply, dynamic memories, paddles, speaker, cassette, etc, all STANDARD. Look at virtually every "personal" computer since. We needed $250,000 to build a thousand--where do you get that kind of money when you're a couple of kids with no business experience? We sought venture money and Mike Markkula agreed to HELP us write a business plan. He realized we were onto something that happens once a decade -- a huge market expanding out of nothing. He joined us (equal partner) and loaned $250,000. He told me I had to quit HP and go 100% Apple. HP is a good company and it's hard to leave any company for anything when you believe it's good to its employees. I said "NO" on my ultimatum day and we were not going to do Apple. Steve Jobs was (in

tears) and got relatives and friends of mine to call me at work and tell me why I should start Apple. Finally I realized I could have a great time doing the one important thing in my life -- design computers for myself and start the company to make money and in my head they didn't have to be dependent. So I turned around. Markkula decided that he and Jobs had better have 52% of Apple combined -- I realize now that they were probably afaid I was a little unpredictable. A true story."

About a faster //e: "The Accelerator [Saturn, 3.58MHz 6502 with 64K RAM] is my favorite card, largely because without any fancy jumpers EVERYTHING ran with it. The only exception with the software I use is Word Juggler under ProDOS. The current Accelerator should have problems with the //e extended memory usage once software uses it. I heard that they are working on a new one to get around this. Its amazing to see everything work faster. My main direction on return to Apple was to get 3.6 MHz built in. Look for it someday. Saturn has shown it's possible."

Apple Assembly Line is published monthly by S-C SOFTWARE CORPORATION, P.O. Box 280300, Dallas, Texas 75228. Phone (214) 324-2050. Subscription rate is $18 per year in the USA, sent Bulk Mail; add $3 for First Class postage in USA, Canada, and Mexico; add $12 postage for other countries. Back issues are available for $1.50 each (other countries add $1 per back issue for postage).